LLVM DISTRIBUTORS CONFERENCE 2021

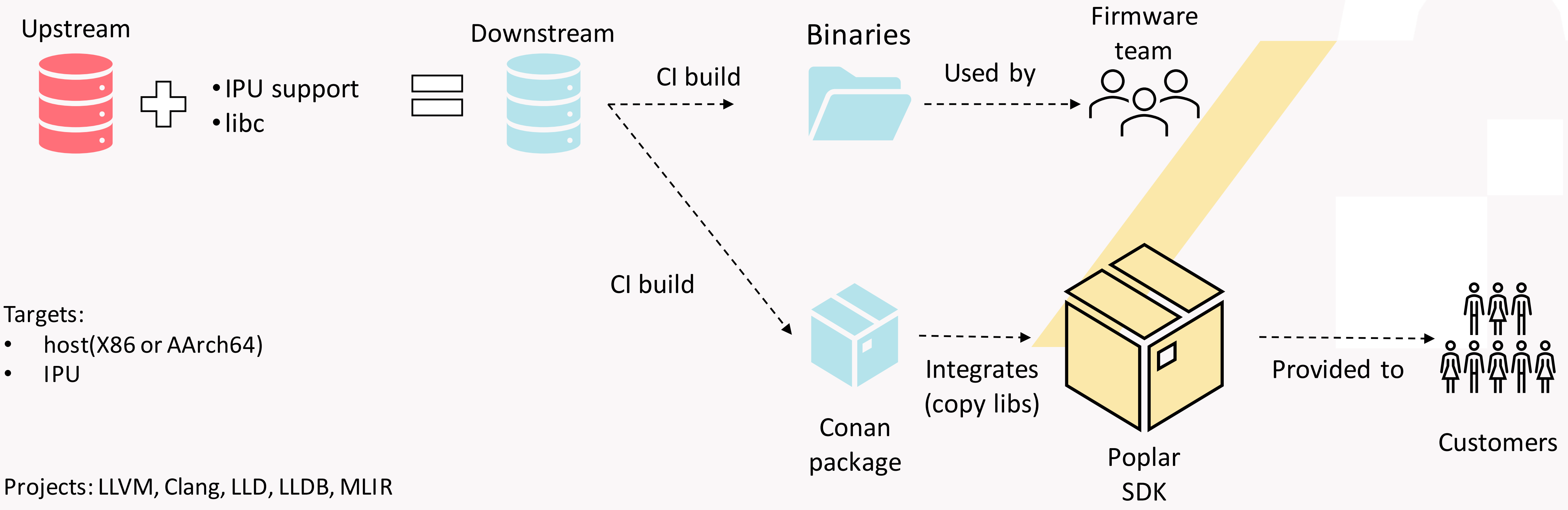
Thomas Preud'homme

GRAPHCORE

Graphcore IPU: quick facts

- Designed to accelerate machine intelligence
- Contains 1472 cores running 6 parallel threads each
- Each core have their own fast local memory for a combined total of 900MB
- Development for IPU using Poplar® SDK
  - uses LLVM to compile for individual cores
  - uses MLIR for some of the high-level optimisation
- More info at https://www.graphcore.ai/products/ipu

Graphcore IPU LLVM distribution



Targets:

- host(X86 or AArch64)
- IPU

Projects: LLVM, Clang, LLD, LLDB, MLIR

External changes

Merges

- Target-specific code in shared files
- Merge process reviewability
- SDAG unit testing
- Behaviour changes in tools
- Ongoing freestanding proposal
- Running tests as root

Fast changing ISA

- Instruction tablegen generation

Target-specific code in shared files

Merges

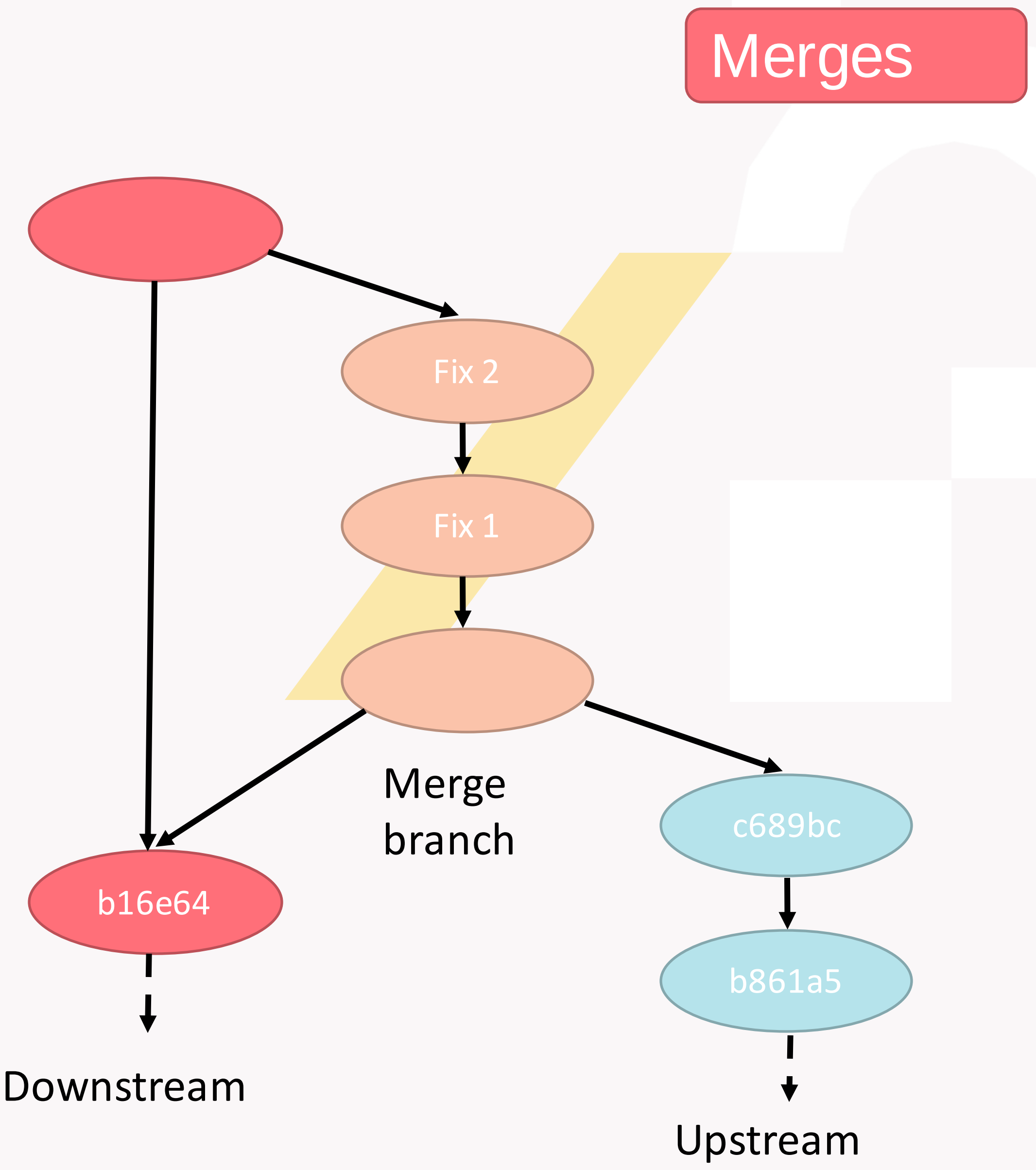
```
TargetInfo *AllocateTarget(const llvm::Triple &Triple,
                           const TargetOptions &Opts) {
  llvm::Triple::OSType os = Triple.getOS();
  switch (Triple.getArch()) {
```

Plenty of examples:

```
clang/include/clang/Basic/Attr.td
clang/include/clang/Basic/AttrDocs.td
clang/include/clang/Basic/TargetBuiltins.h
clang/include/clang/Driver/Options.td
clang/lib/Basic/Targets.cpp
clang/lib/CodeGen/CGBuiltin.cpp
clang/lib/CodeGen/TargetInfo.cpp
clang/lib/Driver/CMakeLists.txt
clang/lib/Driver/Driver.cpp
clang/lib/Driver/ToolChains/Clang.cpp
Clang/lib/Driver/ToolChains/CommonArgs.cpp
Clang/lib/Sema/SemaDeclAttr.cpp
(…)
```

Merge process reviewability

How to review merge-related changes?



SDAG unit testing

Merges

Motivation: SDAG node with no natural IR representation

Example: ANY_EXTEND only appear through combine

Solution: SDAG unit testing

```
; RUN: llvm-link %isdopc %s | opt -instcombine -always-inline | llc

@ISD_ANY_EXTEND = external constant i32
declare i2 @llvm.ipu.SDAG.unary.i4.i2(i32, i2)

define i4 @test(i2 %x) {
  %id = load i32, i32* @ISD_ANY_EXTEND
  %res = call i4 @llvm.ipu.SDAG.unary.i4.i2(i32 %id, i2 %x)
  ret i4 %res
}
```

ISD constants file generation

Merges

ISDOpcodeConstantsPrinter.cpp:

```
const StringMap<std::pair<StringRef, int>> ISDNameConstantMapping = {
  MAP(ISD_X, ISD::X),
  (…)
}

for (auto &Mapping : ISDNameConstantMapping) {
    std::tie(VarName, ISDValue) = Mapping.getValue();
    std::cout << "extern constexpr unsigned " << VarName.str() << " = "
              << ISDValue << ';' << std::endl;
}
```

CMakeLists.txt:

1. Build ISD printer program with **host** compiler
2. Run program and compile output with target compiler

Behaviour changes in tools

Merges

Example: llvm-lit's --no-indirectly-run-check

Our setup: Lit invoked on individual tests by CMake in many repositories

Problem:

- Lit now errors out if a test would not have been run if invoked on parent directory
- Lit from both internal and external LLVM distribution

Solution 1: New commit to add lit config option to control it

Solution 2: Use lit from pip

Ongoing freestanding proposal

Merges

Context:

- no dynamic allocation on IPU for efficiency and robustness reasons
- current C++ specification for freestanding implementation requires dynamic allocation

Problem: Very invasive diff to add support

Our approach: copy libc++ in separate repo, allowing less frequent merge schedule

Note: current status of ongoing freestanding proposal no longer contains dynamic allocation

Running tests as root

Merges

Context: builds and tests run in docker as root

Problem: access right tests

Examples:

- `llvm/test/tools/llvm-ar/error-opening-permission.test`
- `llvm/test/tools/llvm-dwarfdump/X86/output.s`
- `llvm/test/tools/llvm-ifs/fail-file-write.test`

Tablegen generation

Fast changing ISA

Motivation: Software-hardware co-design requires quick support of new instructions

Solution: Generation of instruction tablegen files & tests from ISA specification

```
class inst_x_aaan : Instruction, Sched<[Res]> {

  let AsmString = "not $op0, $op1";
  dag OutOperandList = (outs AR:$op0);
  dag InOperandList = (ins AR:$op1);
  (…)
}
```

And if not manually defined:

```
def X : inst_x_aaan;
```

Pros:

- Freely updated assembler
- Allow mix of manual and automated definitions

Cons:

- Not suitable for upstream

THANK YOU

Thomas Preud'homme
thomasp@graphcore.ai